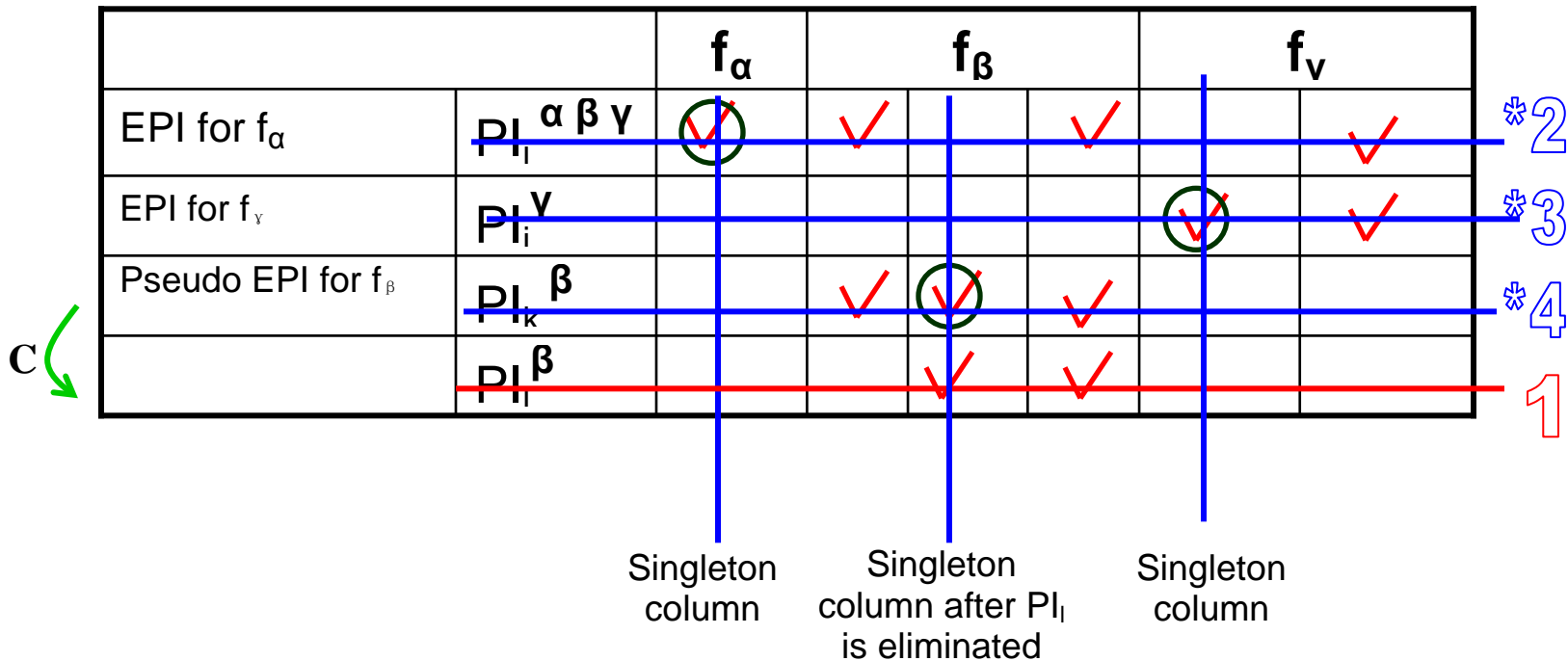


ECE 465 Midterm Solutions, S'09

Shantanu Dutt

1 (a)



In the above example, $PI_i^{\alpha\beta\gamma}$ is an EPI for f_α . Further, all the MTs it covers in f_β and f_γ are covered by other PIs, PI_k^β and PI_i^γ , that are EPIs or pseudo-EPIs for f_β and f_γ , respectively. Thus it is not necessary to include $PI_i^{\alpha\beta\gamma}$ in f_β 's and f_γ 's expressions (although there is no hardware cost for having $PI_i^{\alpha\beta\gamma}$ in their expressions). If we include $PI_i^{\alpha\beta\gamma}$ in the expressions for f_β and f_γ , that would be cases of *unnecessary sharing* of a PI ($PI_i^{\alpha\beta\gamma}$).

(b)

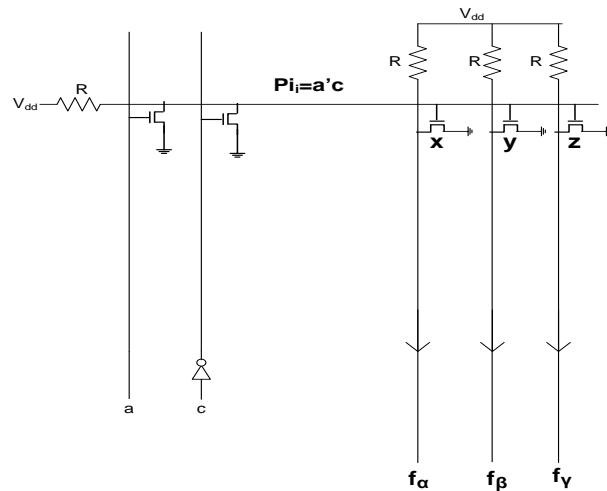


Figure 1.2 PLA programming without using rule 7

(c)

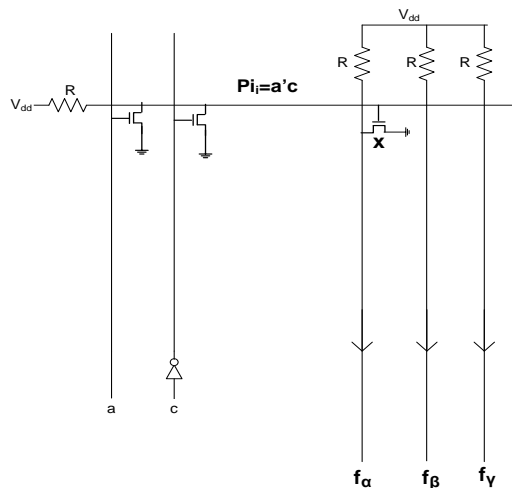


Figure 1.3: PLA programming using rule 7

Continuing the previous example, we saw that using $PI_i^{a\beta\gamma}$ in f_β and f_γ is a case of unnecessary PI sharing. However, if we do that, then the PI_i line (as an example, PI_i is shown here to be $a'c$) needs to drive an extra two transistors gates (transistors y and z) compared to not including PI_i in f_β and f_γ . It will thus increase the capacitor loading on the PI_i line and its delay will thus increase (delay of a PI line = $R * C_{total}$, where C_{total} is the total capacitor load on the PI line).

As shown by the example in part (b), unnecessary PI sharing results in increased delays on PI lines of such PIs (those that are unnecessarily shared). Application of Rule 7 can avoid unnecessary PI sharing, since rule 7 allows a PI to be used in a function only when it becomes an essential or pseudo-essential PI or when it is used to break a cyclic PI. Rule 7 can thereby keep delay on each multi-function PI lines to be a minimum, as these lines will now only drive transistors in the OR plane/array for functions in which it is only included by necessity.

It the above example, PI_i will not be included in f_β and f_γ if rule 7 is used and the delay of its line will reduce compared to the PLA programming shown in Figure 1.2. Figure 1.3 shows the new PLA programming using rule 7.

2 (b)

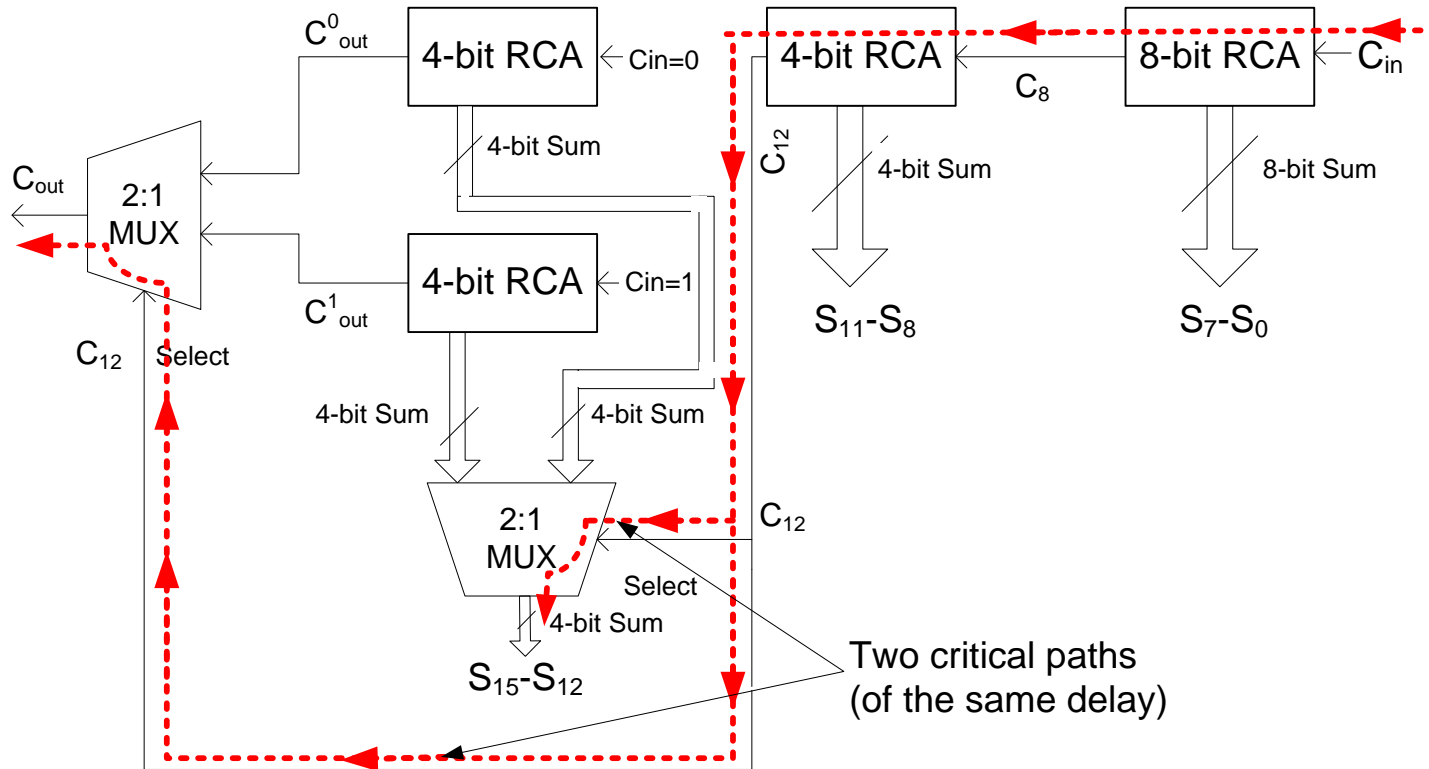


Figure 2.2: Adder circuit

(c)

As shown in Figure 2.2, a 12-bit RCA of the LS 12-bits and the two 4-bit RCAs of the MS 4-bits perform in parallel. Therefore, their overall delay will be the maximum of these two modules' delays. In this case, the 12-bit RCA will have a higher delay and their overall delay will be $12 \times 10 = 120$ nsec. The critical path is the carry chain path (C_{in} to C_{out}) through all 12 FAs of the 12-bit RCA, and each FA imposes 10 nsec delay.

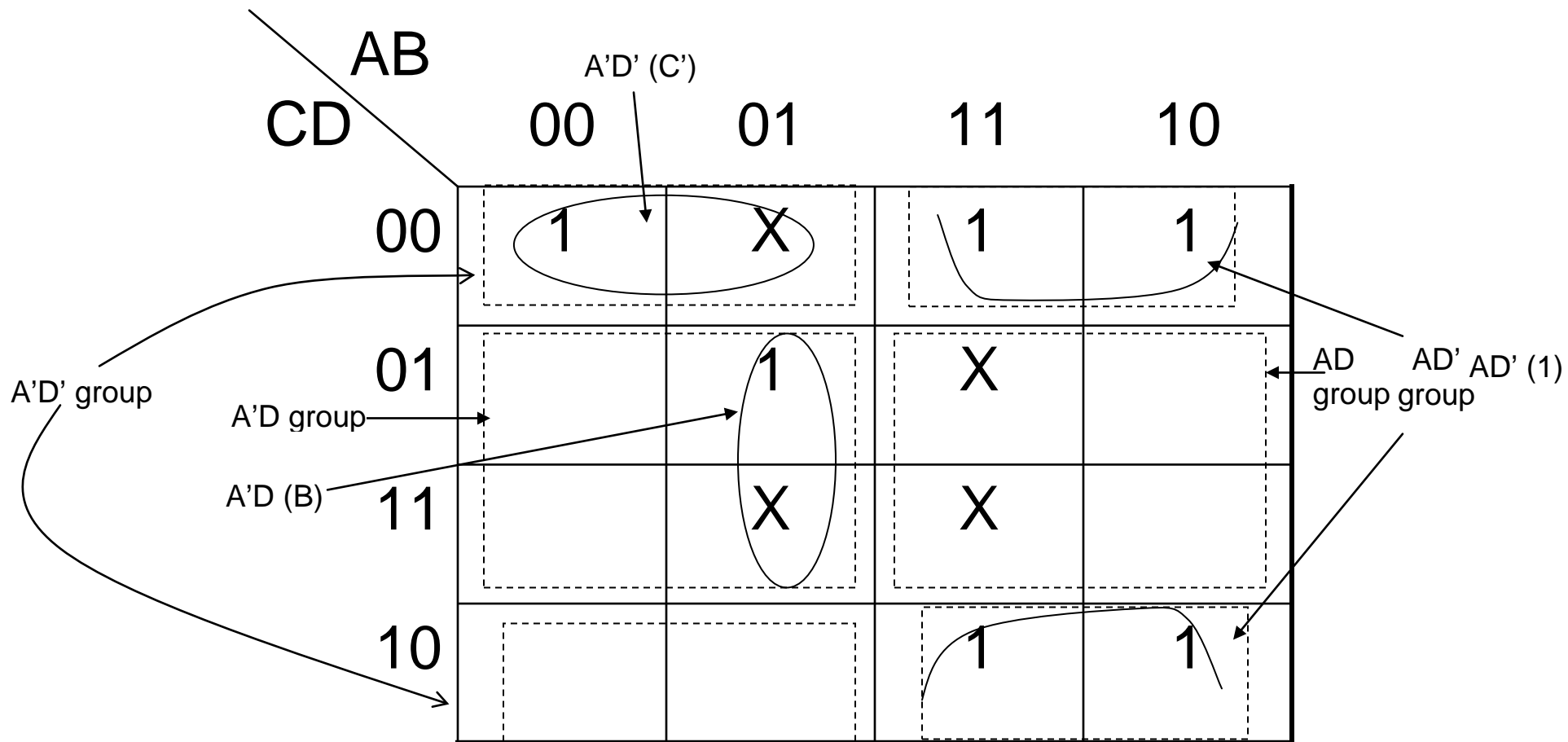
After the 12-bit RCA completes, the two MUXs (for the Sum and Cout bits) get their select inputs from the carry-out bit C_{12} generated by the 12-bit RCA, and take an additional 2 nsecs to connect their appropriate inputs to their outputs (both the MUXes process their inputs in parallel)

Thus the total delay of the above *partially speculative (i.e., partial carry-select) adder* = 120+2 = 122 nsec.

(d) 16-bit RCA delay = 16*10=160 nsec. Again, the critical path goes via the carry chain, through all FAs.
Thus, the partially speculative 16-bit adder delay is less than that of a 16-bit RCA.

3.

$$f(A, B, C, D) = \sum_m (0,5,8,10,12,14) + \sum_d (4,7,13,15)$$



By observation, choosing A and D as select signals seems to yield the lowest cost logic across the four AD groups.

$$f(A, B, C, D) = \bar{A}\bar{D}(\bar{C}) + \bar{A}D(B) + A\bar{D}(1) + AD(0)$$

Note that the above solution is *not unique* in terms of minterm cost. Other select variables (e.g., C,D) could also give a min-cost implementation.

